a first object representation associated with an object;

persistence information associated with said first object representation;

event information relating to interaction with said object,

said object being instantiated on a first machine and being output from the first machine to a second machine, wherein after said object is output from said first machine, the instantiation of said object is deleted from the first machine.

REMARKS

Reconsideration and entry of the present amendment are respectfully requested. In the prior Action, claims 1-9 were pending. Claims 1-9 remain pending and no new claims are added.

Claim 5 has been amended to place the claim in better form for appeal. No new matter is added.

Claim 1-2 and 5-7 stand rejected under 35 U.S.C. 102(e) as being anticipated by Dale et al. (U.S. Patent No 6,272,673, filed November 25, 1997, issued August 7, 2001). Applicants respectfully traverse.

Claim 1 recites a system for transporting objects between a first and second machine where the first machine is programmed in a first language and the second machine is programmed in a second language, the system comprising, among other things, a first processor on the first machine for executing code and instantiating an object on the first machine and an output for outputting the object with persistence information to the second machine wherein after the object is output from the first machine, a first

processor deletes the instantiation of the object from the first machine. Dale does not teach each and every limitation of the claim 1 invention.

The Office Action and Final Office Action ("Action") alternately assigns various elements as supposedly equivalent to the elements in the claim 1 invention. However, the argument presented in the Action lacks consistency by failing to assign any particular element in the Dale disclosure as being equivalent to a given element in the claim 1 invention. The Action picks and chooses a different "equivalent" within any one examination of the elements of the claim 1. However, an organized comparison of the elements of the Dale disclosure against the claim 1 invention, on the other hand, reveals that Dale fails to teach each and every aspect of claim 1.

The Action asserts that Fig. 3 of Dale supposedly teaches the claim 1 invention, however, this assertion is incorrect. Referring to Fig. 3 of the Dale disclosure, the Action asserts that the server 24a of Fig. 3 is equivalent to the "first machine" and the client 20a is equivalent to the "second machine". Even assuming this comparison is valid, claim 1 recites, in addition to the first machine and the second machine, a first processor on said first machine for executing said code and instantiating an object on said first machine and an output for outputting said object with persistence information to said second machine. Therefore, following the Action's line of reasoning, in order for the rejection to be valid, Dale would have to disclose in Fig. 3 a first processor on server 24a for executing code and instantiating an object on server 24a and an output for outputting the object with persistence information to the client 20a (the element that the Action is equating with the second machine).

However, it is clear that Dale does not provide this teaching in Fig. 3. The Action equates "component 64" with the "object on said first machine", however, Fig. 3 does not illustrate "component 64". Perusal of the entire Dale reference reveals that the Action has switched in the middle of the analysis to Fig. 8B where "component 64" is illustrated presumably because Fig. 3 fails to teach each and every element of claim 1.

Changing the focus of analysis to Fig. 8B of the Dale reference likewise does not teach or suggest claim 1 as the Action suggests. Presumably, the Action continues to equate the first machine of claim 1 with the server 24a and the second machine of claim 1 with the client 20a. As in the analysis of Fig. 3, in order for an anticipatory rejection to be valid assuming the assumptions made in the Action, Dale would have to disclose a first processor on server 24a for executing code and instantiating an object on server 24a and an output for outputting the object with persistence information to the client 20a. Even assuming that "component 64" illustrated in Fig. 8 of Dale is equivalent to the "object" recited in claim 1 as the Action asserts, Dale still does not teach or suggest claim 1. Fig. 8B is described at Col. 12 of the Dale reference. Component 64 is not instantiated as stored in storage facility 27 (col. 12, lines 13-14). The application server 24a of Dale causes component 64 to be instantiated and executed on the application server 24a (col. 12, lines 21-22). However, following the Action's line of reasoning that "component 64" is equivalent to the "object" and that "client 20a" is equivalent to the "second machine", Dale does not teach or suggest outputting "component 64" with persistence information to the client 20a. However, using the Action's arguments and assumptions of equivalent elements, such a teaching would be necessary to sustain a valid anticipatory rejection. As

is evident in Fig. 8B, after component 64 is instantiated on server 24a, component 64 is not disclosed as being output anywhere, much less to the client 20a.

Analyzing each of the other components (namely, component 62, component 63, or component 65) that are illustrated in Fig. 8B and described at Col. 12 in turn makes it clear that none of the other components permit the Dale reference to satisfy the requirement of teaching each and every element of claim 1.

In regards to component 63 illustrated in Fig. 8B, the application server 24a (the element equated with the *first machine* by the prior Action) causes component 63 to be instantiated and executed on the client 20a (the element equated with the *second machine* by the prior Action). Hence, if component 63 is equated with the "object" as the Action implies, Dale still fails to teach or suggest a first processor on the first machine (server 24a) for executing said code and instantiating an object (component 63) on the first machine (server 24a). Even assuming the assumptions made by the Action are valid, Dale would merely disclose the server 24a (first machine) executing code and instantiating component 63 (an object) on the client 20a (*second machine*, not the first machine).

In regards to component 65 illustrated in Fig. 8B, the application server 24a (the element equated with the *first machine* by the prior Action) causes component 65 to be instantiated and executed on another server 66. Hence, if component 65 is equated with the "object" and the server 24a is equated with the first machine as the Action implies, Dale still fails to teach or suggest a first processor on the first machine (i.e., server 24a) for executing said code and instantiating an object (i.e., component 65) on the first machine (server 24a).

In regards to component 62 illustrated in Fig. 8B, the Action asserts that component 62 is equivalent to the "object". However, this assumption is incorrect. Fig 8B illustrates component 62 as an HTML page. The application server 24a (equated with the first machine by the Action) provides the HTML page 62 (equated with the object by the Action) to the client 20a (equated with the second machine by the Action). In order for a valid anticipatory rejection using "component 62" as the "object", the server 24a as the first machine and the client 20a as the second machine as the Action attempts to accomplish, Dale would be required to disclose a first processor on the server 24a (first machine) for executing said code and instantiating component/HTML page 62 on the server 24a and outputting the component/HTML page 62 with persistence information to the client 20a. However, Dale merely discloses the server 24a providing the HTML page 62 to the client 20a. Clearly, even assuming that component 62 is the "object", Dale still does not teach or suggest claim 1.

Moreoever, Dale does not teach or suggest after said object is output from said first machine, said first processor deletes the instantiation of said object from said first machine. The Action cites Dale at col. 13, lines 52-55 for this teaching. As explained in Applicant's prior response, Dale merely describes a component being explicitly destroyed. However, claim 1 recites not merely that the object is deleted but that "after the object is output from the first machine, the first processor deletes the instantiation of the object from the first machine." Dale does not teach or suggest "after the object is output from the first machine". Dale does not teach or suggest the object being output from the first machine at all. The Final Action does not respond to this deficiency in Dale.

The Action has improperly applied the Dale reference to claim 1 by randomly choosing elements in Dale to equate with each of the recitations of the object in claim 1. In the first instance of an "object" (the second step in claim 1), the Action asserts that component 64 is supposedly equivalent. In the second instance of the object (the third step in claim 1) the Action then asserts that it is not component 64 that is the equivalent anymore but rather component 62. It is respectfully pointed out that the second step of claim 1 recites "an object" and the third step of claim 1 recites "*said* object". The Action does not address the fact that component 64 does not satisfy the third step of claim 1 and component 62 does not satisfy the second step of claim 1. Nor does the Action address the fact that neither component 64 nor component 62 satisfy the recitation that "object is output from said first machine."

The Final Action asserts that, "Fig. 3 of Dale reference shows arrows going back and forth between the client and the server, which clearly provides the limitation 'an output for outputting the object'." Final Action page 6, lines 8-10. However, contrary to the Action's assertions, the arrow in Fig. 3 does not teach or suggest an output for outputting said object with persistence information to said second machine and after said object is output from said first machine, said first processor deletes the instantiation of said object from said first machine. The Action equates the second machine with client 20a, the first machine with server 24a and the object with component 64 in the second step of claim 1 (and alternately with component 62 in the third step of claim 1). Clearly, the arrow in Fig. 3 does not demonstrate an output for outputting either component 64 or component 62 to the client 20a from the server 24a and after component 64/62 is output

from the server 24a, said first processor (on the server 24a) deletes the instantiation of component 64/62 from the server 24a.

The Final Action attempts to address the lack of a first processor by asserting that, "Dale clearly shows in Fig. 2 the architecture of a computer system with a processor 10. Dale further states that any computer systems mentioned in the invention may have the architecture of Fig. 2 (lines 24-42 column 4)." However, claim 1 recites that a first processor on said first machine for executing said code and instantiating an object on said first machine. Therefore, even if Dale discloses a "processor" in a computer (in Fig. 2), it is still insufficient to teach or suggest claim 1 which specifically recites a first processor on said first machine for executing said code and instantiating an object on said first machine. Dale does not teach or suggest the processor in Fig. 2 instantiating an object on the first machine as set forth above, for example. Thus, the generic "processor" disclosed by Dale is not equivalent to the first processor in claim 1.

The Final Office Action responds to the lack of disclosure of persistence information in the Dale reference by asserting that "...Dale reference clearly points out the issue as explained in the office action and again in 'include information such as designation ...' (lines 28-49 column 9)." Final Office Action page 6, lines 17-19. The Final Office Action further asserts that, "applet tag ... is not used to make equivalent to persistence information [in the prior office action]... the information 'include information such as designation ...' (lines 28-49 column 9) within the applet tag was used to meet the limitation [of persistence information]." Final Office Action page 7, lines 13-15. The "designation" information to which the Action refers is information contained in an applet tag that indicates on which tier the component is to be executed and the duration of

the instantiation on the tier (col. 9, lines 27-31). However, claim 1 recites "instantiating an object on the first machine" and "outputting said object with persistence information to the second machine". If the "designation" information of Dale is the same as the persistence information of claim 1 as the Office Action purports, then claim 1 would recite instantiating an object on the first machine and outputting the object with a "designation as to where to instantiate the object" (equated with "persistence information" by the Office Action) to the second machine. This would not make sense because the object has already been instantiated. Therefore, contrary to the assertions of the Final Office Action, Dale does not teach or suggest persistence information or outputting the object with persistence information to the second machine with the disclosure of "designation" information.

The Final Office Action responds to the lack of teaching of "wherein after said object is output from said first machine, said first processor deletes the instantiation of said object from said first machine" by stating "the applicant is advised to refer back to the office action where the limitation is clearly met by Dale reference." Final Office Action page 6, lines 21-22. Applicants have referred back to the office action as the Final Office Action suggests, however, it was not shown in the prior office action or the Final office action that the limitation is "clearly met by Dale reference" as the Action purports.

As set forth above, Dale merely discloses "unregistration of a component may occur ... when the component becomes no longer instantiated, such as if the component is explicitly destroyed." Dale, Col. 13, lines 53-55. However, claim 1 recites "after said object is output from said first machine, said first processor deletes the instantiation of said object from said first machine." Dale does not teach or suggest "after said object is

output from said first machine" because Dale does not teach or suggest outputting said object from said first machine at all. Even assuming that the "unregistration" of Dale's "component" is the same as deleting the instantiation of the object, Dale still does not teach or suggest claim 1 because Dale at best only discloses merely deleting a component. Mere deletion of a component is insufficient. Dale does not teach or suggest after the object is output from the first machine, the first processor deletes the instantiation of the object from the first machine nor does the prior office action (or the final office action) address this issue.

For at least the above reasons, it is respectfully submitted that the rejection of claim 1 is improper and should be withdrawn.

Claim 5 as amended and claim 6 are similar to claim 1 and are allowable for at least the reasons set forth above for claim 1.

Claim 2 depends from claim 1 and claim 7 depends from claim 6. Therefore, the rejections of claims 2 and 7 should be withdrawn for at least the reasons set forth above for claim 1.

Claim 5 recites a data structure for allowing the interchange of objects between a server and a client comprising a first object representation, persistence information associated with said first object representation and event information relating to interaction with said object.

Claims 3 and 8 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dale in view of Chang (U.S. Patent No. 5,960,436). Applicants respectfully traverse.

Claim 3 depends from claim 1. As set forth above, Dale does not teach or suggest claim 1. Chang does not make up for the deficiencies of Dale. Claims 1 and 3 recite a

system for transporting objects between a first and second machine comprising, among other things, a first processor on the first machine for executing code and instantiating an object on the first machine and an output for outputting the object with persistence information to the second machine wherein after the object is output from the first machine, a first processor deletes the instantiation of the object from the first machine.

Chang discloses that a client requests objects from a server, downloads the objects needed, stores the objects in a storage location in a local file system, then disconnects from the server and executes transactions locally (col. 2, lines 1-6). The transactions are recorded by the client's local file system and when the client reconnects with the server, the transactions are sent to the server and replayed at the server to modify objects at the server, the modified objects being written back to the server's database. However, Chang does not teach or suggest an output for outputting the object with persistence information to the second machine. Chang merely discloses a client performing transactions and sending the transactions to a server. The Final Office Action responds to the lack of teaching of outputting events or objects with persistence information by stating "Chang was not used to teach outputting events with persistence information ... Chang was used to teach passing the modified objects back to the server as clearly pointed out [in] the claim rejection." Claim 3 recites "the first machine reinstantiates the objects based on the persistence information and handles the events as effecting the reinstantiated objects." Claim 3 does not recite "passing the modified objects back to the server" as the Action purports. Therefore, even assuming that Chang teaches "passing the modified objects back to the server", Chang would still not teach or suggest the claim 3 invention. However, Chang does not teach or suggest "passing the modified objects back to the

server" as the Action purports. Chang merely sends *transactions* to a server and not "modified objects". In any event, Chang does not teach or suggest claim 1 from which claim 3 depends. Therefore, neither Dale nor Chang, either alone or in combination, teach or suggest claim 3 and the rejection should be withdrawn.

Claim 8 was rejected for the same reasons as claim 3. Claim 8 is similar to claim 3 and is allowable for at least the reasons set forth above for claim 3. Therefore, the rejection of claim 8 should also be withdrawn.

Claims 4 and 9 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dale in view of Barlow (U.S. Patent Number 6,275,935) and Chang. This rejection is respectfully traversed.

Claim 4 recites an input in said first machine for receiving persistence information and an event from said second machine, a processor in said first machine for instantiating an object based in part on said persistence information, and an event handler in said first machine for handling said event in combination with modifying said object.

The Final Office Action responds to the lack of teaching in Dale of the first machine for receiving persistence information and an event from the second machine by asserting, "Fig. 3 ... shows arrows going back and forth between the client and the server, which clearly provides the limitation of 'an input for receiving' ... Dale further teachings 'include information such as designation ...' (lines 28-49 column 9), which meets the persistence information limitation." However, as set forth above, the "designation" information merely describes which tier the component is to be executed on and the duration of instantiation. Therefore, the "designation" information of Dale is not

equivalent to persistence information and one of skill in the art would not mistakenly confuse the two concepts.

The Final Office Action responds to the lack of teaching in Dale of receiving an event from the second machine by asserting, "Dale also teaches the component 26 contains the properties such as event that can be scripted through point-and-click dialogs (lines 46-61 column 6) which clearly meets the event from the server." However, claim 4 recites an input in said first machine for receiving an event from said second machine, an event handler in said first machine for handling said event in combination with modifying said object, and an output for outputting said modified object to said second machine. Thus, the input in the first machine is for receiving an event from the *second machine*. Also, an output for outputting the modified object to the second machine. Even assuming the component 26 contains an event, Dale still does not teach or suggest a first machine with an input for receiving the event from the second machine. The Action has equated the client 20a with the first machine. If this is the case, even if the client 20a receives the "event", the client 20a still does not receive the "event" from the second machine (equated with server 24a in the Office Action). Nor can any element in Dale be equated with the "second machine" of claim 4 because there is not teaching or suggestion in Dale for outputting the modified object to any element disclosed in Dale.

The Final Office Action responds to the lack of teaching or suggestion in Barlow of modifying the object by stating that, "Barlow was not used to teach modifying the object ... Barlow was used to teach an event handler that handles the event ... *Chang* is the reference that actually teaches modified the object and send it to the second machine." Final Office Action, page 8, lines 17-20. Thus, the Final Office Action now admits that

Barlow does not teach or suggest modifying the object. However, contrary to the Action's assertions, Chang does not make up for the deficits. Chang does not teach or suggest modifying "the object and send it to the second machine" as purported in the Final Office Action. Rather, Chang merely performs transactions locally while disconnected, then connecting and sending a log of transactions to the server where the log is replayed on the server. The resulting modified objects are stored in memory at the server. Col. 2, lines 1-13. Notably, Chang sends a log of transactions but does not send modified objects.

Claim 9 was rejected for the same reasons as claim 4. Claim 9 is similar to claim 4 and is allowable for at least the reasons set forth above for claim 4. Therefore, the rejection of claim 9 should also be withdrawn.

Applicants respectfully submit that the instant application is in condition for allowance. If the Examiner feels, however, that further amendment and/or discussion may be helpful in facilitating prosecution of the case, the Examiner is respectfully requested to telephone the undersigned attorney of record at the number appearing below.

Respectfully submitted,

Christopher R. Glembocki
Registration No. 38,800

BANNER & WITCOFF, LTD.
1001 G Street, N.W.
Washington, D.C. 20001
(202) 508-9100

Date: December 13, 2002

MARKED-UP VERSION OF AMENDMENTS

In the claims:

Please amend claims as follows (a marked-up version of the amendments is attached in the Appendix):

5. (Amended) a data structure for allowing the interchange of objects between a server and a client comprising:

a first object representation associated with an object;

persistence information associated with said first object representation;

event information relating to interaction with said object,

said object being instantiated on a first machine and being output from the first machine to a second machine, wherein after said object is output from said first machine, the instantiation of said object is deleted from the first machine.